

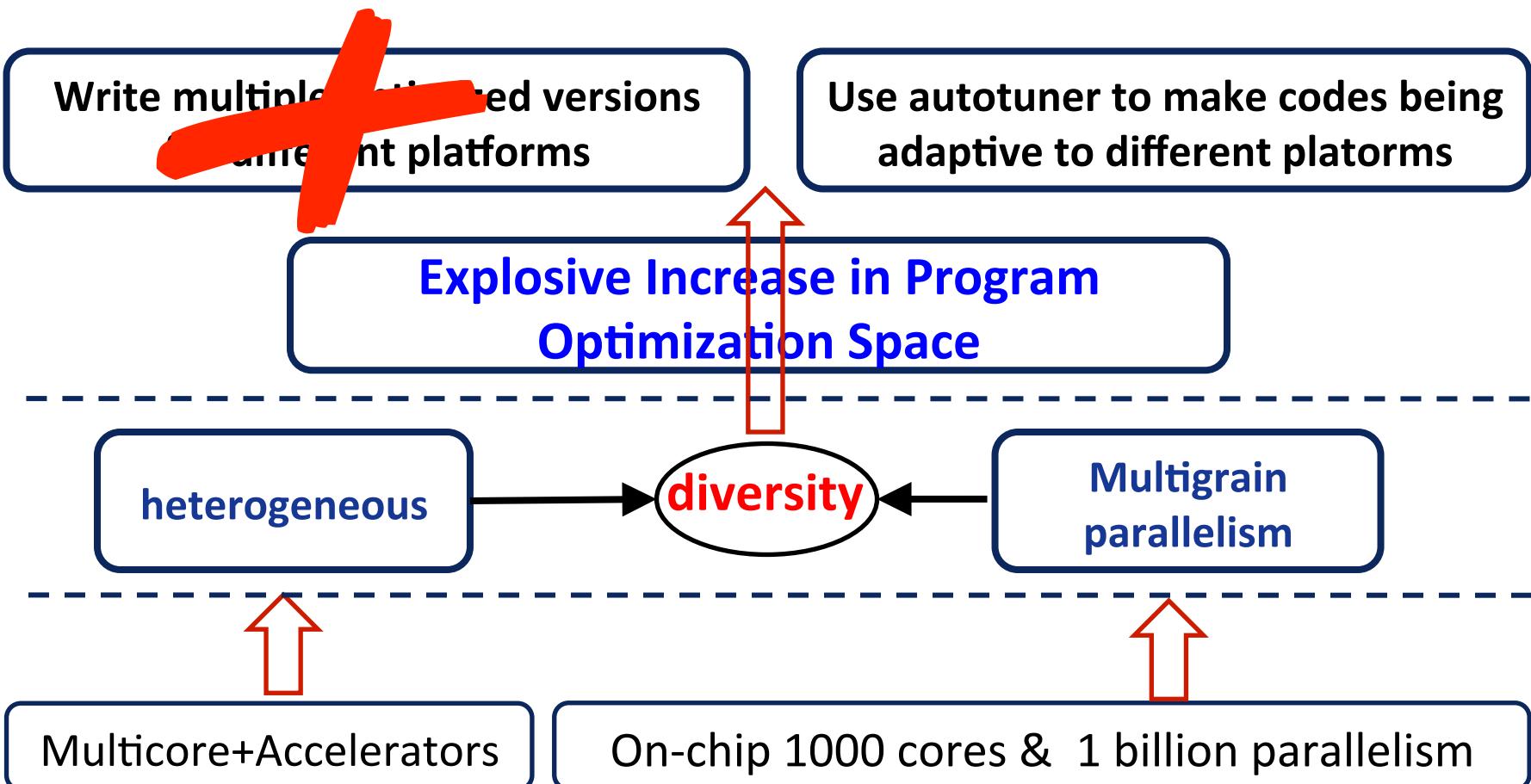
An Autotuning Protocol to Rapidly Build Autotuners

Guangming Tan

Institute of Computing Technology, Chinese Academy of Sciences

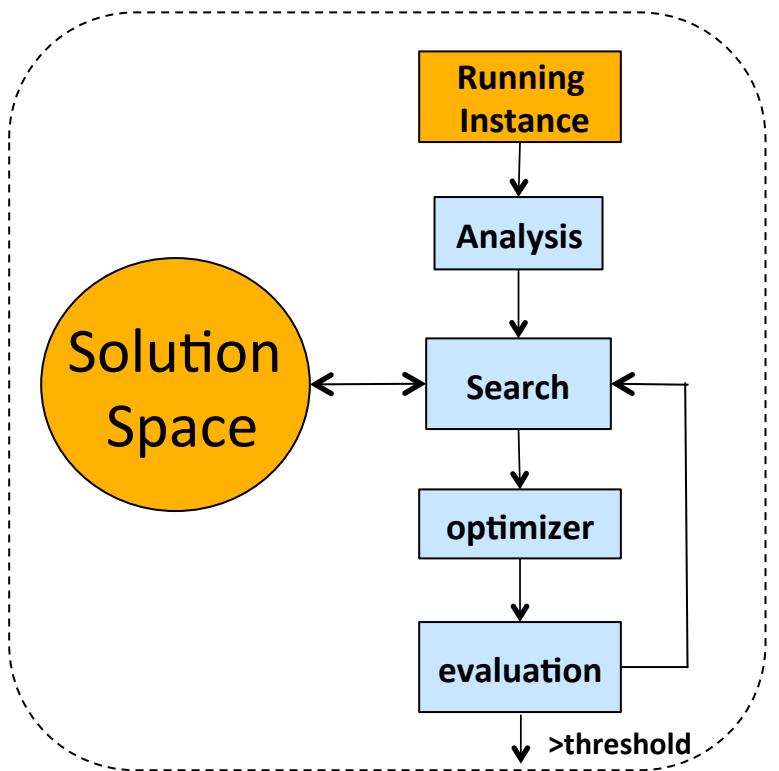
2015.11.15

Performance Challenge in Exascale Computing



Exascale System

Key Issues of Current Autotuners



- All steps are closely coupled so that we have to rewrite the whole autotuner for updating implementation of any step
 - No **composable** modules
 - No **resuable** moduels

Autotuning Hierarchy

Solutions

Stencil autotuner

SpMV autotuner

.....

FFT autotuner

?

Tools

ORIO, HPS, OpenTuner,
TAU, PAPI, gprof, ...

.....

Algorithms

Exhaustion search	Decision tree
Greedy search	Clustering
Heuristic search	Deep Learning

.....

Problems

Platforms

Single-node: CPU/GPU/MIC/FPGA/
Multi-nodes: CPU/CPU+X

.....

Domains

Stencil, SpMV, FFT, Graph

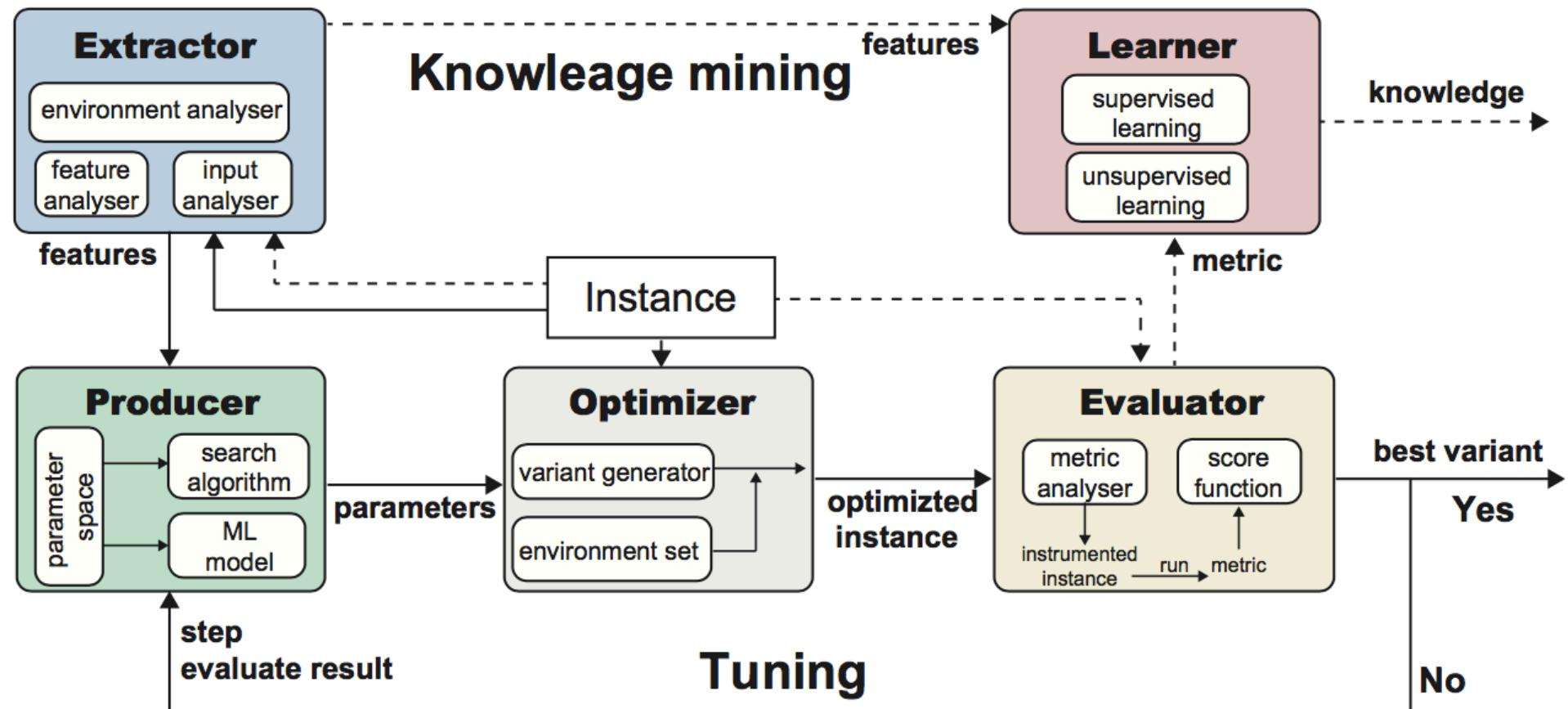
.....

Environments

OS, memory hierarchy,
runtime library, compiler

.....

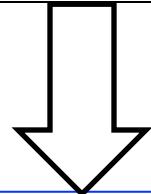
PAK: an Autotuning Protocol & Infrastructure



- Infrastructure to provide autotuning tools
- Composable & reusable modules

Extractor

Running Instances



- program code
- input
- target platform
- back-end compiler
- ...

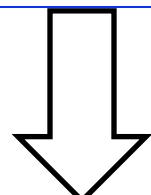
```
#customize an Extractor object 'ext' using a  
set of analyser objects
```

```
ext=PAK.Extractor$new(analysers)
```

```
#invoke extractFeatures() and extract  
features from an application
```

```
ext$extractFeatures(app)
```

List of Features



<Feature>

Name: reuse, flops, #arrays, ...

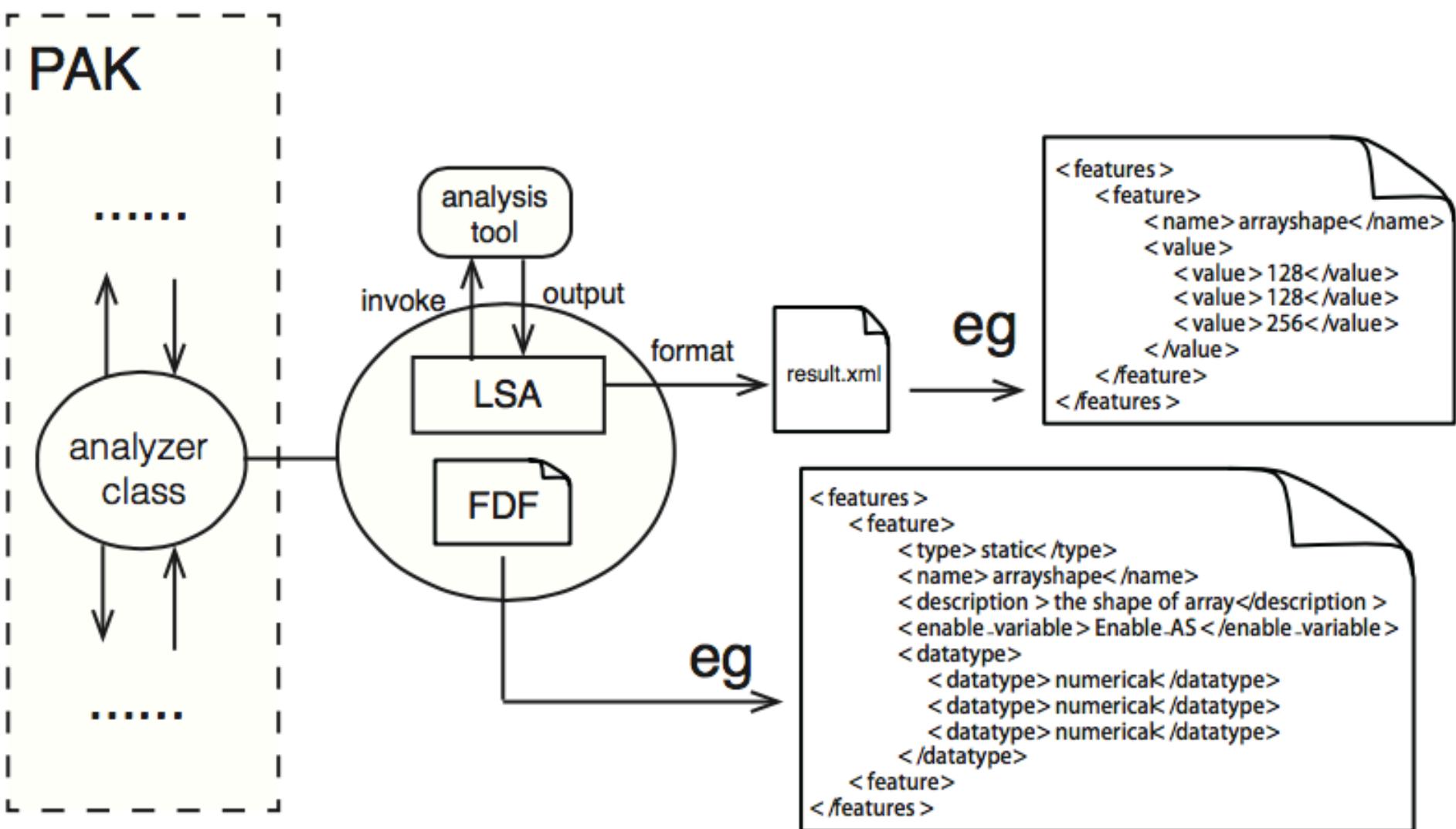
Description:

Datatype:

Enable_variable:

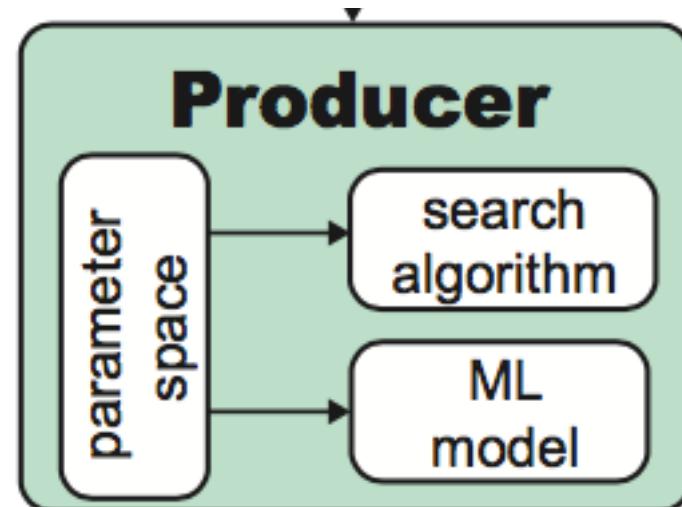
XML </Feature>

Customized Analyser



Producer

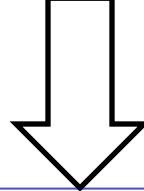
```
# an exhaustion search Producer class
PAK.Producer.Exhaustion<-setRefClass("PAK.Producer.Exhaustion",
contains="PAK.Producer",
fields = list(parameter.space="data.frame"),
methods = list(
  #init function
  initialize=function(parameter.space){
    parameter.space<<-parameter.space      },
  #implemente the interface method
  getParameter=function(step,extractor.result,score)      {
    if(step<nrow(parameter.space))
    return(parameter.space[step,])
    else
    return (NULL)
  }
)
)
```



Optimizer

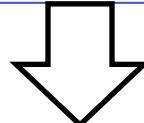
Optimization Strategies

- Parameterized compiler
- highly-tuned library
- ...



```
#customize an Optimizer object 'opt'  
uses the hpsGen[1]  
opt=PAK.Optimizer$new("hpsGen")
```

```
#instances the optimization strategies  
on the running instance  
opt$optimize(app,parameters)
```



Optimized Running Instance

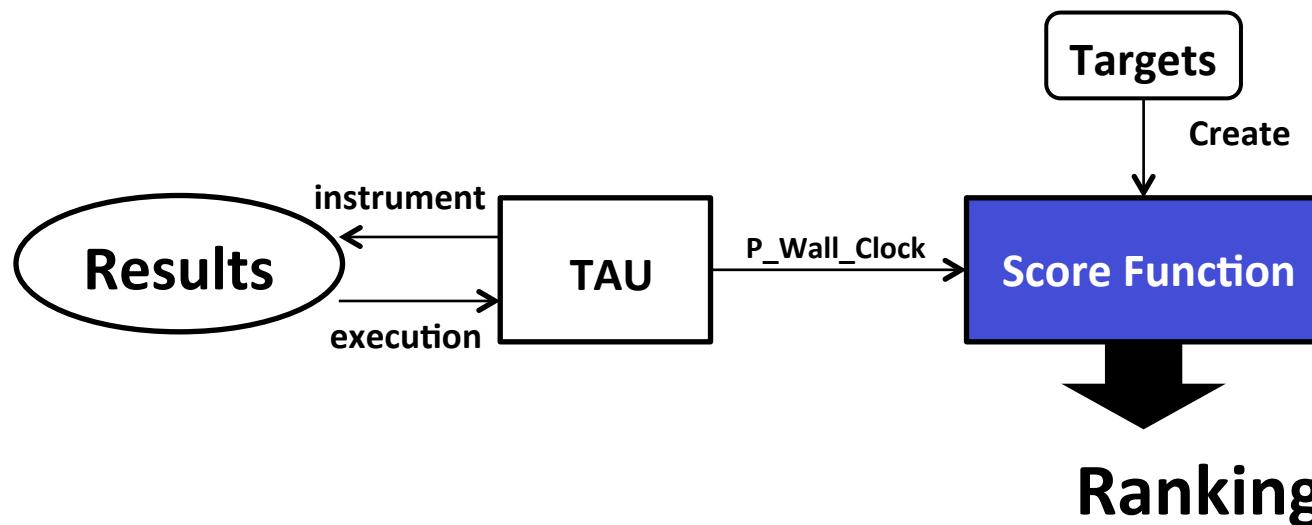
Evaluator

```
#define two score functions
s1=function(x){if(x>100) return (100-x) else return(0)}
s2=function(x){if(x>10^7) return (10^7-x) else return(0)}

#wraps the analyzers with the score functions
subevaluators=list(tau=list(P_WALL_CLOCK_TIME=s1,
                             PAPI_LST_INS=s2))

#customize an Evaluator object 'eva' using the analyzers
eva=PAK.Evaluator$new(subevaluators)

# evaluate an running instance and return the result score
score=eva$evaluate(optimized.instance)
```

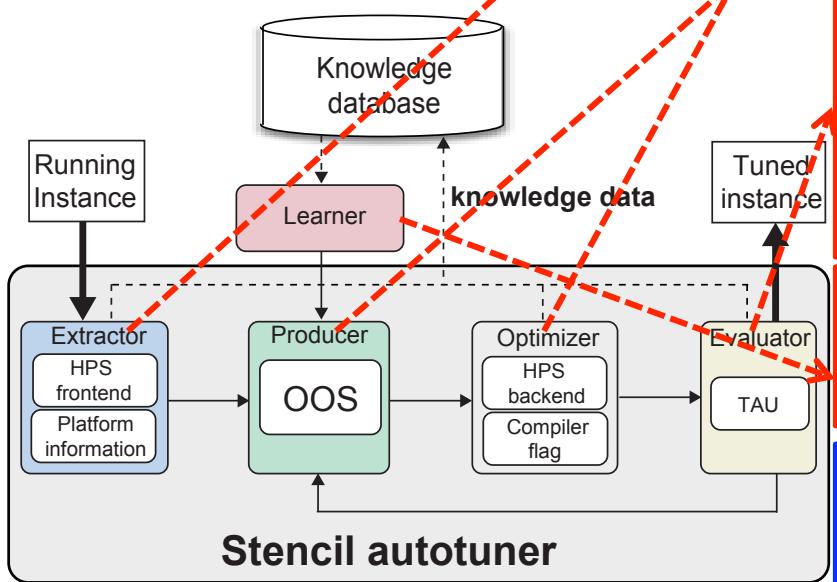


Learner

Machine Learning Toolkit

```
# an decision tree Learner Class
PAK.Learner.DecisionTree<-setRefClass("PAK.Learner.DecisionTree",
contains="PAK.Learner",
  fields = list(model="list",dv.name="character",
idv.name="character"),
methods = list(
  #init function
  initialize=function(){      },
  learnModel=function(training.data,idv,dv){
    buildstr<-sprintf("rpart(%s~.,training.data)",dv)
model[["rp"]]<<-eval(parse(text=buildstr) )
model[["dv.name"]]<<-dv
model[["idv.name"]]<<-idv
return (model)
}
)
)
```

Stencil Autotuner



```

myextract<-C.Extracto
r
$new(list(hps_frontend=c("FD","RD","AN","LR","AC","DT","PS",
"IT"),platforminfo=c("core","freq","cache","bd")))

myoptimizer<-C.Optimizer$new(generator.name="hps_gen")

#training
myproducer<-C.Producer.Exhaustion$new(parameter.list)

myevaluator<-C.Evaluato
r
$new(sub.evaluators=list(tau=list(P_WALL_CLOCK_TIME=function
(x){if(x>0) return (-x) else return(0)})))
for(app in training.stencils){ tuning<-C.Tune
r
$new(app=app,optimizer=myoptimizer,evaluator=myevaluator,pro
ducer =myproducer,need.store=TRUE) tuning$tune()}

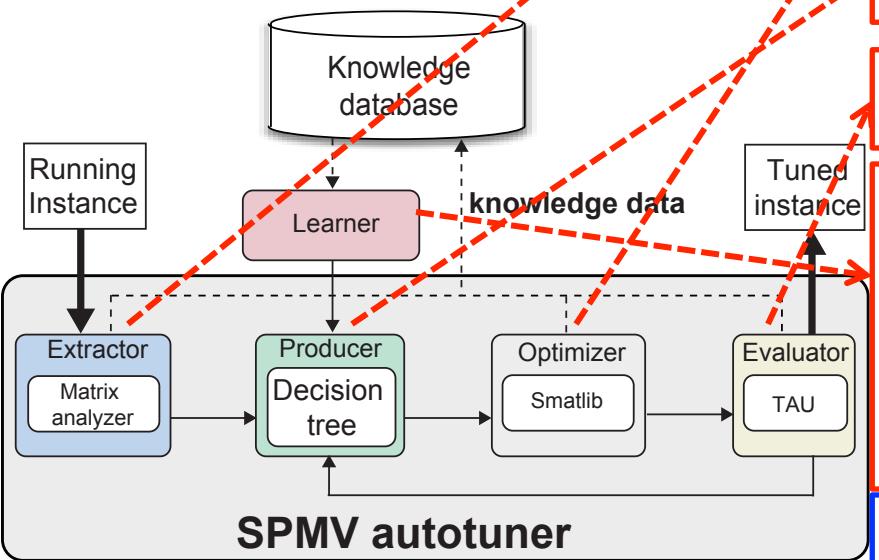
#learn model
mylearner<-C.Learner.OptimalSpac
e$new(trainingData)myproducer<-C.Producer.OptimalSpac
e$new(mylearner$learn())

#autotuning using oos model [1]
for(app in test.stencils){
t<-targettime[[app]]
myevaluator<-C.Evaluato
r
$new(sub.evaluators=list(tau=list(P_WALL_CLOCK_TIME=function
(x){if(x>t)return (-x) else return(0)})))
tuning<-C.Tune
r
$new(app=app,extractor=myextract,optimizer=myoptimizer,evalu
ator=myevaluator,producer =myproducer)

tuning$tune()
}

```

SpMV Autotuner



```

#customize an Extractor object 'ext' and an Optimizer object
'opt'
ext=PAK.Extracto
r
$new(list(spmv_extractor=c("M","N","NNZ","Ndiags","NTdiags_ratio
","ER_DIA")))

opt=PAK.Optimizer$new(generator.name="smatlib")

#customize an Producer object 'prol' using exhaustion algorithm
and an Evaluator object 'eva' prol=PAK.Producer.Exhaustio
n$new(list(method_name='MV_COO','MV_CSR','MV_CSC','MV_DIA';))

eva=PAK.Evaluato
r$new(sub.evaluators=list(tau=list(P_WALL_CLOCK_TIME=function(x
){if(x>0) return (0-x) else return(0)})))

#perform autotuning for training
for(app in training.matrixs){ tuning=PAK.Tune
r$new(app=app,optimizer=opt,evaluator=eva,producer
=prol,need.store=TRUE)
tuning$tune()

#learn the model
pro2=PAK.Producer.DecisionTree$new()
pro2$trainModel(trainingData,"","method_name")

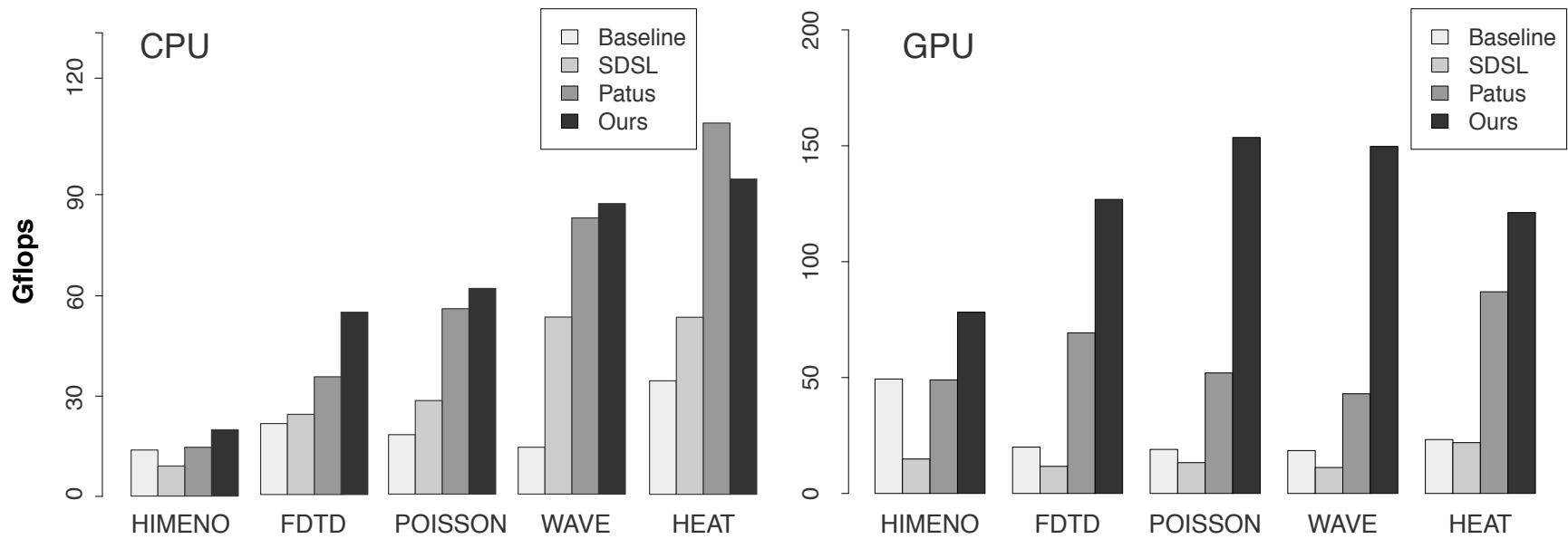
#autotuning using generated decision-tree model
for(app in test.matrixs){
t=objectives[[app]] eva2=PAK.Evaluato
r$new(sub.evaluators=list(tau=list(P_WALL_CLOCK_TIME=function(x
){if(x>t) return (t-x) else return(0)})))
    tuning=PAK.Tune
r$new(app=app,extractor=ext,optimizer=opt,evaluator=eva,producer
=pro2)
    tuning$tune()
}

```

Re-implementation of SMAT autotuner [2]

[2]. Jiajia Li, Guangming Tan, Mingyu Chen, Ninghui Sun:
SMAT: an input adaptive auto-tuner for sparse matrix-vector
multiplication. PLDI 2013: 117-126

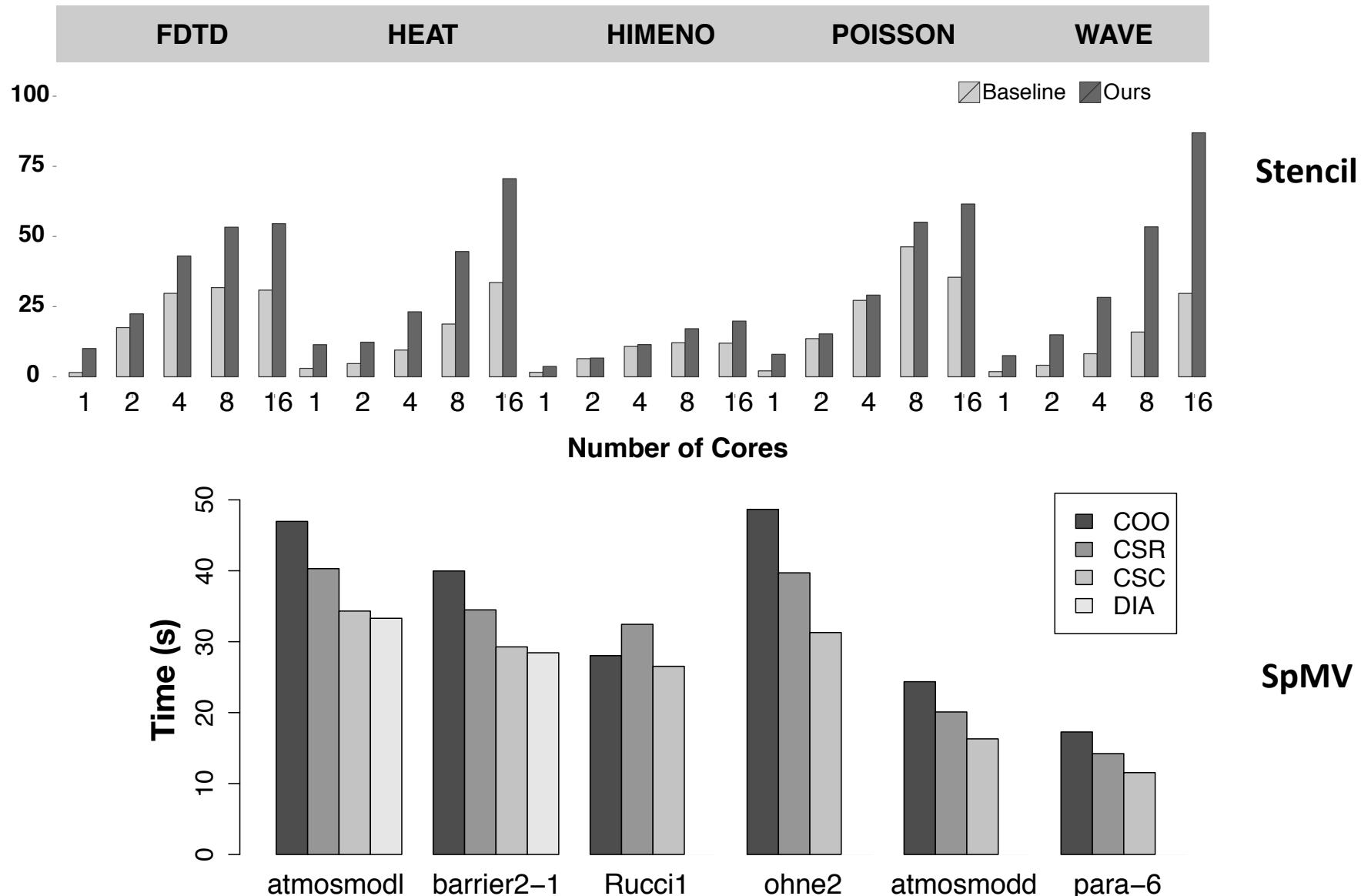
Performance Comparison (Stencil)



	CPU	GPU
Baseline	328%	486%
SDSL	100%	830%
PATUS	18%	125%



Performance



Conclusion

- We propose a protocol to rapidly build an autotuner
 - extractor, producer, optimizer, evaluator, learner
- We demonstrate its usage with two Exascale computing kernels
 - Stencil
 - SpMV
- Open Source
 - <https://github.com/luoyulong/PAK>

Thanks!